Exhibit 6

DOCUMENT SOUGHT TO BE SEALED

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

ORACLE AMERICA, INC.,	)	
	)	
Plaintiff,	)	
	)	
v.	)	Civ. A. No. 10-03561 WHA
	)	
GOOGLE INC.,	)	*(Jury)*
	)	
Defendant.	)	

REPLY EXPERT REPORT OF PROFESSOR DOUGLAS C. SCHMIDT, Ph.D.

February 29, 2016

programming. A search of the Udacity website for "renderscript android" results in zero hits. It is reasonable to conclude that because Google's own sanctioned training for Android app development does not focus on RenderScript, Google itself must not consider RenderScript relevant for learning to program Android apps.[42]

C. Dr. Astrachan Continues to Overstate the Connection Between the Java Programming Language and the 37 Java Packages

70. In his Rebuttal Report, Dr. Astrachan continues to claim that "the Java programming language is inextricably intertwined with the use of underlying class libraries."[43] This sentiment is echoed in Dr. Leonard's report, which constantly conflates the Java programming language with the APIs at issue.[44]

71. I have reviewed TX1062, which is described as "classes and interfaces mentioned in the Java Language Specification (third edition) (excluding examples and commentary)." While the classes listed in TX1062 are, in fact, mentioned in the Java Language Specification (3rd ed.) ("JLS"), very little declaring code or SSO is specified in the JLS. In other words, even if I were to accept the premise that code elements mentioned in the JLS were necessary (or "inextricably intertwined") with the Java programming language, very little declaring code or SSO is actually mentioned. For example, as I have described previously, classes in the 37 Java API Packages include numerous methods, constructors, fields, or other elements. In most instances, the JLS simply requires the *presence* of a class of a particular name; the JLS does not specify nor require the constituent parts of that class, nor the relationships between classes and interfaces.

MOOCs (*see* http://www.nytimes.com/2012/11/04/education/edlife/the-big-three-mooc-providers.html?_r=0), currently boasting over 1.6 million subscribed users (*see* http://www.fastcompany.com/3021473/udacity-sebastian-thrun-uphill-climb), with a more recent particular focus on vocational courses for professionals, which are called "Nanodegrees." *See* https://www.udacity.com/nanodegrees-new-s/nd801 for information on the Android Nanodegree that Udacity co-created with Google.

[42] The unimportance of RenderScript is further highlighted by a statement made by a senior engineer on stackoverflow: *"You can see from the lack of answers that Renderscript is hardly used outside of Google because of its strange architecture that ignores industry standards like OpenCL and almost non-existent documentation on how it actually works."* http://stackoverflow.com/questions/21774211/render-script-rendering-is-much-slower-than-opengl-rendering-on-android.

[43] Astrachan Rebuttal Report ¶104.

[44] *See, e.g.*, Leonard Report ¶100.

72. I understand that Dr. Reinhold has reviewed whether any declaring code or SSO is subject to a technical constraint imposed by the JLS and has made a list of any such code. I have reviewed and independently verified Dr. Reinhold's list. To the extent these declarations appear in Exhibit S of Mr. Zeidman's Opening Report, these declarations would be subject to a technical constraint imposed by the JLS. (Conversely, copied lines of code not present on the list are not constrained by the JLS.) Thus, even if one were to assume that elements of the 37 Java API Packages mentioned in the JLS are necessary to the language (which I do not agree with), it affects very little of what Google has copied in this case. Appendix H includes Dr. Reinhold's list of constrained declarations from the 61 classes of TX1062. Appendix I shows the copied SSO, removing the relationships described in the JLS.

D. Dr. Astrachan Mischaracterizes Certain Elements of the Android Compiler and Runtime Environment

73. In his Rebuttal Report, Dr. Astrachan argues that the dex compiler is not, in fact, a compiler. (Astrachan Rebuttal Report, §IV-D-1). I disagree with this assertion since the Android dx tool comments specifically reference compilation activity. The author comment lines, excerpted below, read:

> dalvik/dx/src/com/android/dx/command/dexer/Main.java:1269
> /** Whether to print statistics to stdout at end of *compile cycle* */
> dalvik/dx/src/com/android/dx/dex/cf/CfOptions.java:44
> /** whether to print statistics to stdout at end of *compile cycle* */[45]
> (emphasis added)

74. Dr. Astrachan's rebuttal report also claims that "ART is not a virtual machine because it does not do runtime interpretation of bytecode." (Astrachan Rebuttal Report, §154) I disagree with this claim based on my analysis of the ART source code, documentation, and tutorials presented on ART by Google engineers. First, ART contains functionality that *does* perform runtime interpretation of bytecode, which resides in the art/runtime/interpreter folder in Android releases K, L, M, and also exists in the Android master branch.[46] Moreover, the Android documentation describes how to generate bytecode for the ART interpreter, where it explicitly describes how/when the interpreter is enabled.[47]

[45] *See* https://android.googlesource.com/platform/dalvik/+/master/dx/src/com/android/dx /dex/cf/CfOptions.java.

[46] *See* https://android.googlesource.com/platform/art/+/master/runtime/interpreter/.

[47] *See* https://source.android.com/devices/tech/dalvik/configure.html#compiler_filtersm.

> causes porting problems... Unfortunately, porting problems will always be with us. The trick is to minimize the problems so developers can concentrate on the application.[52]

82. One article by a single author with a provocative title is not the equivalent of "criticism from the programming community," which is a community that amounts to millions of developers.[53] More critically, however, the article author's main point seems to be that "taking advantage of VM-specific features will cause porting problems," which of course is correct, but misses the efforts put into "Write Once, Run Anywhere", evidenced by the fact that Sun and Oracle have, for more than 20 years, published portability guides and other tools to help developers avoid these VM-specific problems. In fact, Sun and Oracle have done exactly what the author recommends: minimize the problems so that developers can concentrate on application development. While it is hard to achieve complete portability, to use flawless portability as a binary threshold of evaluation clearly misses the point that programming is a creative, expressive art involving numerous choices, tradeoffs, and complex decisions. Moreover, it also misrepresents the fact that the bulk of Java applications port smoothly between Java implementations that have successfully passed the JCK test suites.

IX. ATTESTATIONS

83. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed on this 29th day of February, 2016, in Nashville, TN.

Douglas C. Schmidt

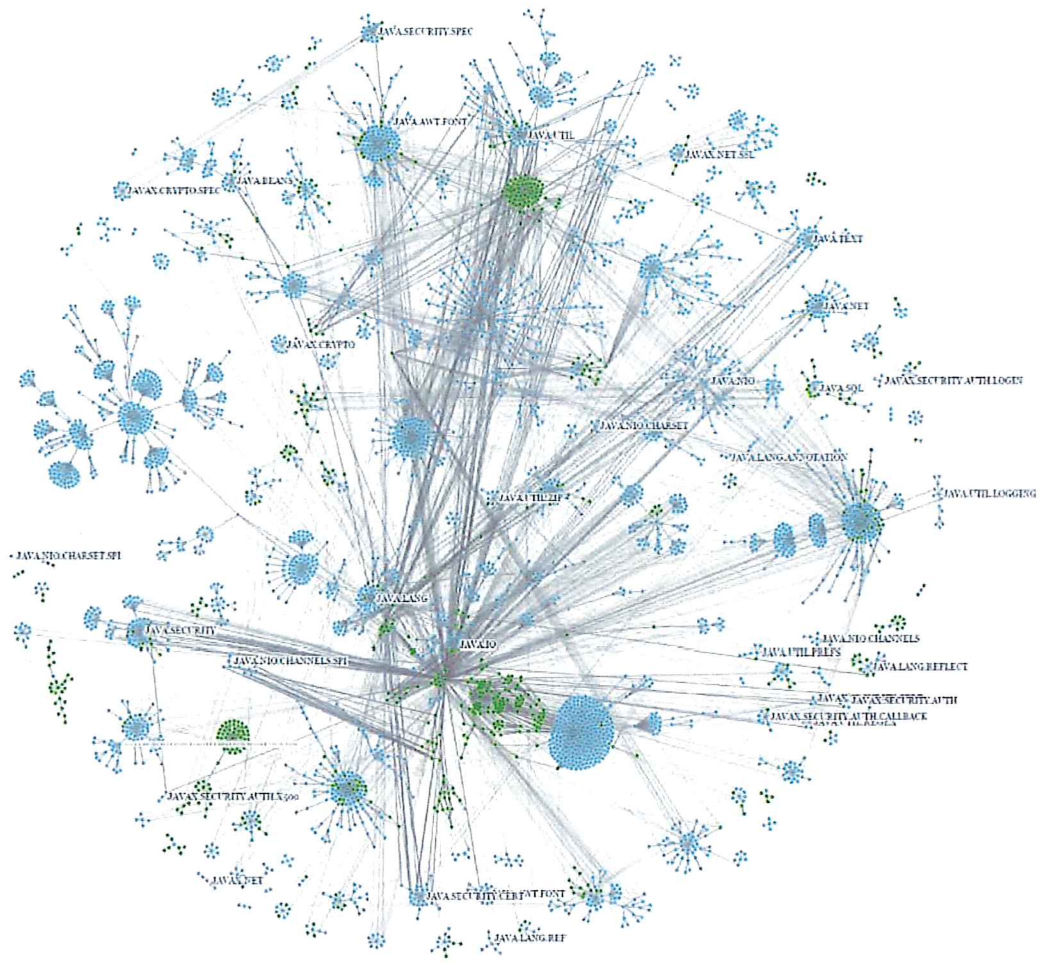
Douglas C. Schmidt, Ph.D.

[52] *Ibid.*

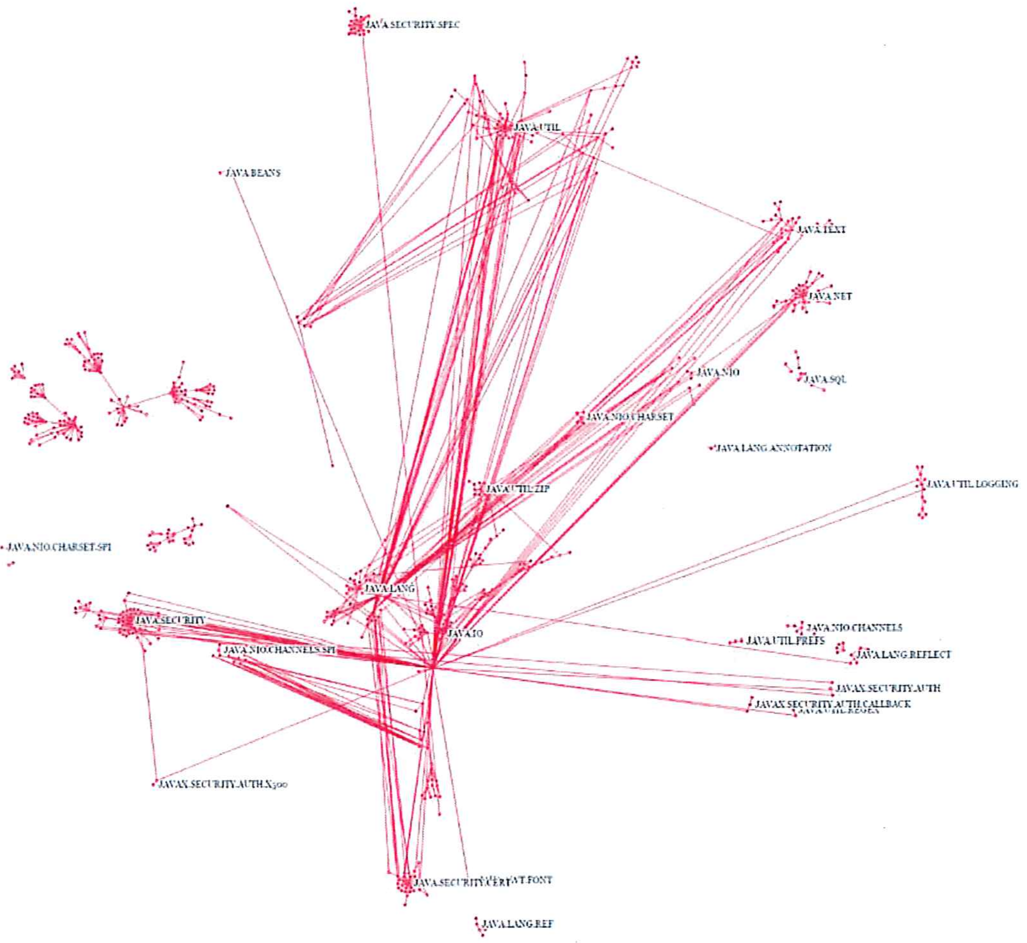
[53] There are 9 Million Java developers worldwide according to https://www.java.com/en/about/.

APPENDIX I: VISUALIZATION OF JAVA SE 5.0 WITH COPIED NODES AND PATHS IN RED, REMOVING THE RELATIONSHIPS DESCRIBED IN THE JLS

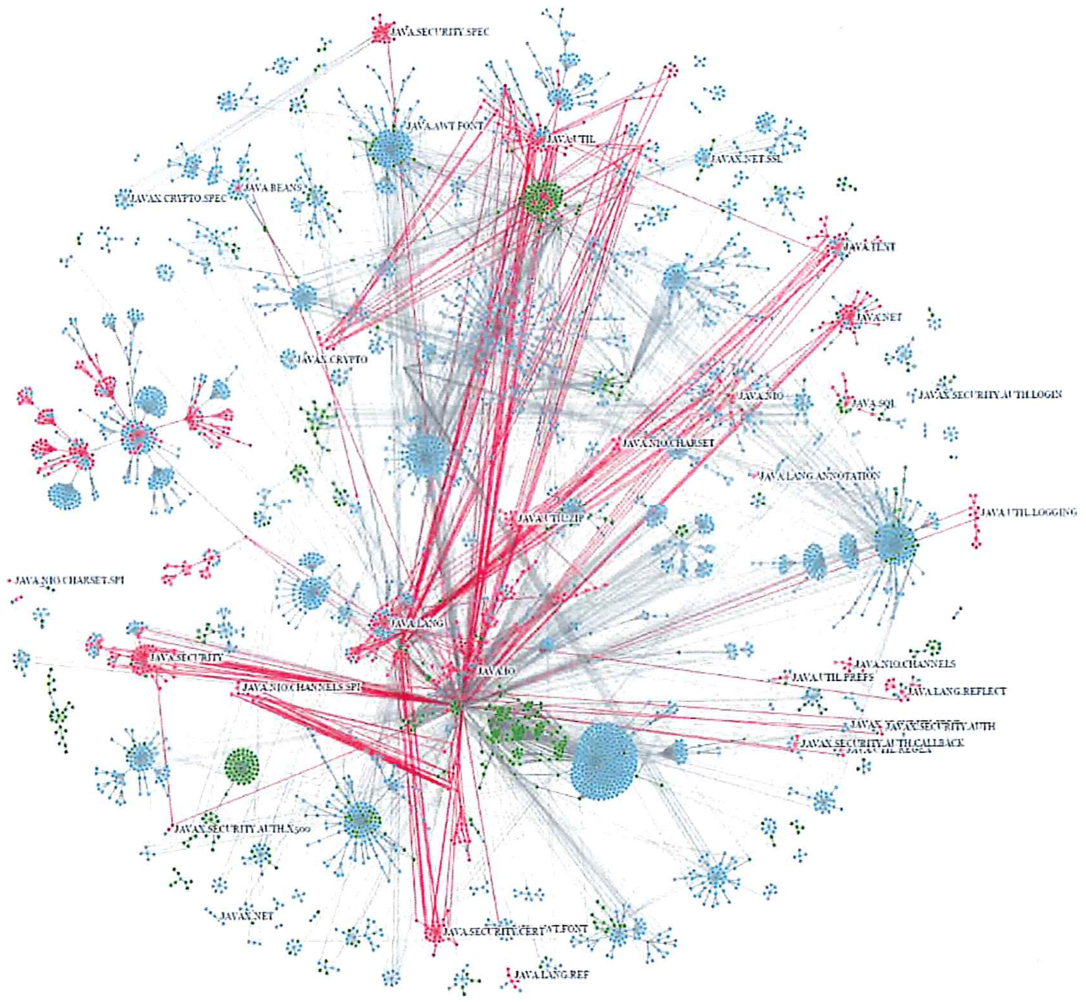
Visualization of the SSO of the 166 Packages in Java SE 5.0



The SSO of the 37 Java API Packages Google Copied in Android Lollipop With Copied Nodes and Paths in Red, removing the relationships described in the JLS (based on copied classes from Mr. Zeidman Opening Report Exhbit S and Appendix H)



The SSO of Java SE 5, with Copied Nodes and Paths in Red


JAVA.SECURITY.SPEC
JAVA.AWT.FONT
JAVA.UTIL
JAVAX.NET.SSL
JAVA.BEANS
JAVAX.CRYPTO.SPEC
JAVA.TEXT
JAVA.NET
JAVAX.CRYPTO
JAVA.NIO
JAVA.SQL
JAVAX.SECURITY.AUTH.LOGIN
JAVA.NIO.CHARSET
JAVA.LANG.ANNOTATION
JAVA.UTIL.LOGGING
JAVA.UTIL.ZIP
JAVA.NIO.CHARSET.SPI
JAVA.LANG
JAVA.SECURITY
JAVA.IO
JAVA.NIO.CHANNELS
JAVA.UTIL.PREFS
JAVA.NIO.CHANNELS.SPI
JAVA.LANG.REFLECT
JAVAX.SECURITY.AUTH
JAVAX.SECURITY.AUTH.CALLBACK
JAVAX.SECURITY.AUTH.X500
JAVAX.NET
JAVA.SECURITY.CERT
AWT.FONT
JAVA.LANG.REF